

## Task 4

(انا اول مره اذاكر C# في حاجات كتير chatGPT ساعدني فيها)

### \*Question 1: Runtime Environment Analyzer

```
Console.WriteLine($".Net runtime Version {Environment.Version}");
Console.WriteLine($"Operating System
{RuntimeInformation.OSDescription}");
Console.WriteLine($"CPU Architecture
{RuntimeInformation.ProcessArchitecture}");

string runtime = RuntimeInformation.FrameworkDescription;
switch (runtime)
{
    case string r when r.Contains(".NET"):
        Console.WriteLine("Modern .NET Runtime");
        break;

    default:
        Console.WriteLine("Legacy Runtime");
        break;
}
```

---

### Question 2: Feature Toggle System

```
namespace ConsoleApp2
{
    internal class Program
    {
        static readonly Version AppVersion = new Version(2,0);
        static void Main(string[] args)
        {

            Console.WriteLine($"Application Version: {AppVersion}");
            Console.WriteLine();

            bool loginEnabled = true;
            bool exportEnabled = true;
```

```

        bool adminEnabled = false;

        Version loginMinVersion = new Version(1, 0);
        Version exportMinVersion = new Version(2, 0);
        Version adminMinVersion = new Version(3, 0);

        CheckFeature("Login", loginEnabled, loginMinVersion);
        CheckFeature("Export", exportEnabled, exportMinVersion);
        CheckFeature("Admin Panel", adminEnabled, adminMinVersion);
    }

    static void CheckFeature(string featureName, bool enabled, Version
minVersion)
    {
        Console.WriteLine($"Checking {featureName} Feature");
        const string DisabledMessage = "Feature Disabled";
        const string AllowedMessage = "Feature Allowed";
        const string VersionMessage = "Version Not Supported";

        if (!enabled)
        {
            Console.WriteLine(DisabledMessage);
        }
        else
        {
            string result = AppVersion >= minVersion ? AllowedMessage
: VersionMessage;

            Console.WriteLine(result);
        }

        Console.WriteLine();
    }
}

```

---

### Question 3: Number Classification Engine

```

using System;
using System.Collections.Generic;

```

```

class Program
{
    static void Main()
    {
        List<int> numbers = new List<int> { 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        var result = ClassifyNumbers(numbers);

        Console.WriteLine("Even Numbers:");
        PrintList(result.EvenNumbers);

        Console.WriteLine("Odd Numbers:");
        PrintList(result.OddNumbers);

        Console.WriteLine("Prime Numbers:");
        PrintList(result.PrimeNumbers);
    }

    static NumberResult ClassifyNumbers(List<int> numbers)
    {
        NumberResult result = new NumberResult();

        foreach (int number in numbers)
        {
            if (IsEven(number))
                result.EvenNumbers.Add(number);
            else
                result.OddNumbers.Add(number);

            if (IsPrime(number))
                result.PrimeNumbers.Add(number);
        }

        return result;
    }

    static bool IsEven(int number)
    {
        return number % 2 == 0;
    }

    static bool IsPrime(int number)

```

```

{
    if (number < 2)
        return false;

    for (int i = 2; i <= number / 2; i++)
    {
        if (number % i == 0)
            return false;
    }

    return true;
}

static void PrintList(List<int> list)
{
    foreach (int n in list)
        Console.Write(n + " ");
    Console.WriteLine();
}
}

class NumberResult
{
    public List<int> EvenNumbers = new List<int>();
    public List<int> OddNumbers = new List<int>();
    public List<int> PrimeNumbers = new List<int>();
}

```

---

#### Question 4: Memory Behavior Test

```

namespace ConsoleApp2
{
    internal class Program
    {
        static readonly Version AppVersion = new Version(2,0);
        static void Main(string[] args)
        {

            User user = new User { Name = "Abdelrahman" };
            UserSnapshot snapshot = new UserSnapshot { Name =
"Abdelrahman" };

```

```

        Console.WriteLine("Before Normal Call:");
        Console.WriteLine($"Class User: {user.Name}");
        Console.WriteLine($"Struct Snapshot: {snapshot.Name}");
        Console.WriteLine();

        Modify(user, snapshot);

        Console.WriteLine("After Normal Call:");
        Console.WriteLine($"Class User: {user.Name}");
        Console.WriteLine($"Struct Snapshot: {snapshot.Name}");
        Console.WriteLine();

        Modifyref(ref user, ref snapshot);

        Console.WriteLine("After Ref Call:");
        Console.WriteLine($"Class User: {user.Name}");
        Console.WriteLine($"Struct Snapshot: {snapshot.Name}");
    }

    static void Modify(User user, UserSnapshot snapshot)
    {
        user.Name = "Modified Class";
        snapshot.Name = "Modified Struct";
    }

    static void Modifyref(ref User user, ref UserSnapshot snapshot)
    {
        user.Name = "Ref Modified Class";
        snapshot.Name = "Ref Modified Struct";
    }
}

class User
{
    public string Name { get; set; }
}

struct UserSnapshot
{
    public string Name { get; set; }
}
}

```

## What Changed?

في ال Method بتاع Modify

ال Class هابتغير لانه Reference Type

ال Struct مش هابتغير لانه Value Type

في ال Method بتاع ModifyRef

هابتغير الاتنين Class & Struct

## Stack vs Heap

### Stack

- Stores:
    - Value types (Structs)
    - Local variables
    - References (addresses) to objects
  - Fast memory
  - Automatically cleared when method ends
- 

### Heap

- Stores:
    - Objects created from Classes
  - Slower than stack
  - Managed by Garbage Collector
- 

## Question 5: Payment Exception Design

```
using System;

namespace ConsoleApp2
{
    internal class Program
    {
        static void Main(string[] args)
        {
        }
```

```
double balance = 1000;

try
{
    ProcessPayment(balance, 1200);
}

catch (InsufficientBalanceException ex)
{
    Console.WriteLine($"Balance Error: {ex.Message}");
}
catch (PaymentTimeoutException ex)
{
    Console.WriteLine($"Timeout Error: {ex.Message}");
}

catch (Exception ex)
{
    Console.WriteLine($"General Error: {ex.Message}");
}

finally
{
    Console.WriteLine("Payment process finished.");
}

static void ProcessPayment(double balance, double amount)
{
    Random rnd = new Random();

    if (amount > balance)
        throw new InsufficientBalanceException("Not enough
balance.");

    if (rnd.Next(0, 2) == 1)
        throw new PaymentTimeoutException("Payment service
timeout.");

    Console.WriteLine("Payment completed successfully.");
}
}
```

```
class InsufficientBalanceException : Exception
{
    public InsufficientBalanceException(string message) :
base(message)
    {
    }
}

class PaymentTimeoutException : Exception
{
    public PaymentTimeoutException(string message) : base(message)
    {
    }
}
}
```

---

## Part 2

### 1 ( Longest Common Prefix)

```
public class Solution
{
    public string LongestCommonPrefix(string[] strs)
    {
        if (strs == null || strs.Length == 0)
            return "";

        string prefix = strs[0];

        for (int i = 1; i < strs.Length; i++)
```

```

{

    while (strs[i].IndexOf(prefix) != 0)

    {

        if (prefix.Length == 0)

            return "";

        prefix = prefix.Substring(0, prefix.Length - 1);

    }

}

return prefix;

}

}

```

## 2(217. Contains Duplicate)

```

public class Solution {

    public bool ContainsDuplicate(int[] nums) {

        Array.Sort(nums);

        for (int i = 1; i < nums.Length; i++) {

            if (nums[i] == nums[i - 1]) {

                return true;

            }

        }

        return false;

    }

}

```

```
}  
  
}
```

### 3(242. Valid Anagram)

```
public class Solution {  
  
    public bool IsAnagram(string s, string t) {  
  
        if (s.Length != t.Length) return false;  
  
        char[] arrs = s.ToCharArray();  
        char[] arrt = t.ToCharArray();  
  
        Array.Sort(arrS);  
        Array.Sort(arrT);  
  
        for (int i = 0; i < arrs.Length; i++) {  
            if (arrs[i] != arrt[i]) return false;  
        }  
  
        return true;  
    }  
}
```